Módulos didácticos digitales como herramienta de apoyo a la enseñanza y el aprendizaje para el desarrollo del pensamiento computacional

Luciano Graziani[1], Gabriela Anahí Cayú[1], Héctor Luis Vivas[2], Paola Verónica Britos[3]

[1] Estudiante, [Aspectos Legales, Profesionales y Sociales de la Informática], Licenciatura en Sistemas, Universidad Nacional de Río Negro

Universidad Nacional de Río Negro, Sede Atlántica, Viedma, Río Negro
http://www.unrn.edu.ar/

{lgraziani, gcayu}@unrn.edu.ar

[2] Profesor, [Aspectos Legales, Profesionales y Sociales de la Informática], Licenciatura en Sistemas, profesor de la asignatura [Aspectos Legales, Profesionales y Sociales de la Informática]

Universidad Nacional de Río Negro, Sede Atlántica, Viedma, Río Negro
http://www.unrn.edu.ar/

lvivas@unrn.edu.ar

[3] Profesora, [Verificación de Programas y Computabilidad], Licenciatura en Sistemas, director trabajo final [Aspectos Legales, Profesionales y Sociales de la Informática]

Universidad Nacional de Río Negro, Sede Atlántica, Viedma, Río Negro
http://www.unrn.edu.ar/

pbritos@unrn.edu.ar

Abstract. Luego de haber realizado un análisis sobre el sistema educativo actual, se identificaron dos problemas. El primero es que los alumnos no tienen incorporado conceptos del pensamiento computacional como la abstracción, el razonamiento lógico-secuencial o la modularización. El segundo, que existe la necesidad de herramientas computacionales que combinen la programación con la currícula del Sistema Educativo Nacional durante la escolaridad obligatoria. La solución es ofrecer al docente y alumnos una herramienta que combine un diseño atractivo y ayude a solucionar problemas por medio de la programación desarrollando el pensamiento computacional. Esta herramienta será una plataforma educativa llamada "CÓDIMO" compuesta de módulos didácticos que serán diseñados a partir de los requerimientos recabados a docentes de nivel primario y secundario. Se comenzará con una prueba piloto aplicada a los alumnos de séptimo grado de la primaria para las materias de matemática y lengua de la Escuela 296 de Viedma, Río Negro.

Keywords: plataforma educativa, módulo didáctico, conceptos, pensamiento computacional, programación, aprendizaje.

1 Introducción

En 1976 Peter Drucker acuñó el concepto de "Sociedad del Conocimiento" que, a diferencia del de "Sociedad Industrial" (basado en la producción industrial), considera al conocimiento y la tecnología como los elementos de mayor impacto para el desarrollo económico y social de las comunidades.

El sistema educativo, hasta el momento, ha trabajado con un modelo de sociedad industrial, respondiendo a necesidades de una sociedad y de un mercado laboral que están desapareciendo.

> En la década de los ochenta, los cambios experimentados en el terreno de la tecnología informática, así como la articulación de las economías nacionales con los procesos de integración y globalización, se introdujeron en las instituciones de educación superior (encargadas de producir, divulgar y transferir conocimiento) dando lugar a la formación de recursos humanos profesionales, científicos y técnicos (Didriksson, 2008) [4].

Estos cambios en la tecnología informática y en los procesos de integración y globalización han dejado en evidencia que es necesario que el modelo de formación que se venía usando en la educación superior también se deba modificar, si es que se quiere dar respuesta a las necesidades de este nuevo contexto.

La enseñanza de la programación en la escuela tuvo origen hace más de cuarenta años con la incorporación del lenguaje Logo, creado por el equipo del MIT (Laboratorio de Inteligencia Artificial del Instituto Tecnológico de Massachusetts) liderado por Seymour Papert (López García, 2013) [7]. Si bien en la década de los ochenta comenzó a difundirse en las escuelas la enseñanza de Logo, a partir de los noventa, con la aparición de sistemas operativos visuales e intuitivos, así como la proliferación de aplicaciones informáticas "empaquetadas" con fines diversos, este espacio fue ocupado paulatinamente por la enseñanza de TIC (Tecnologías de la Información y la Comunicación). El interés en las clases de Informática pasó de la utilización de las computadoras como herramientas para pensar y resolver problemas (Jonassen, 1997) [5] al manejo de aplicaciones vinculadas principalmente a tareas de oficina (paquetes de ofimática). En la visión de Papert los niños deberían ser capaces de diseñar, crear y expresarse con las nuevas tecnologías (Resnick, 2012) [8]. En este sentido (Levis, 2006) [6] sostiene que "una verdadera alfabetización digital [...] debe ofrecer los elementos básicos para el desarrollo de capacidades que permitan la comprensión y dominio del lenguaje en el que están codificados los programas", para que los alumnos no sean meros consumidores de contenido sino que tengan la posibilidad de crearlo.

El sistema educativo no necesita más contenido sino, particularmente, entornos flexibles que permitan desarrollar las capacidades de autoaprendizaje, creatividad, autonomía, iniciativa y trabajo en equipo.

En respuesta a este cambio, el 12 de agosto de 2015 el Consejo Federal de Educación aprobó la Resolución 263/15, que en su artículo primero establece que "la enseñanza y el aprendizaje de la "Programación" es de importancia estratégica en el Sistema Educativo Nacional durante la escolaridad obligatoria, para fortalecer el desarrollo económico-social de la Nación".

En este sentido, Mitchel Resnick expresa que del mismo modo que el aprendizaje de la lectura no es un fin en sí mismo sino que aprendemos a leer para aprender otras cosas, así también se busca que los niños y adolescentes más que aprender a programar, programen para aprender. (Resnick, 2013) [9].

2 La programación y el pensamiento computacional

Al escribir código, el alumno no sólo aprende conceptos matemáticos e informáticos como organizar un proceso, reconocer rutinas o repeticiones y descubrir errores en su pensamiento computacional cuando su programa no funciona según la idea o expectativa con la que lo concibió, también aprende otras habilidades estratégicas como resolver problemas, diseño de proyectos, comunicación de ideas, capacidad de abstracción y planificación, descomposición de problemas, trabajo en equipo, creatividad, entre otras. Todas éstas son características claves del pensamiento computacional.

La programación, en cualquier lenguaje, requiere seamos ordenados en todo momento, de lo contrario, la computadora no será capaz de comprender correctamente nuestras órdenes. Además, al estar escribiendo en un idioma completamente diferente, se ejercita constantemente la capacidad de comprensión para poder plasmar nuestra idea en código. También ejercitamos esta capacidad de comprensión cuando surge un error, pues necesitamos interpretar y comprender los datos que nos proporciona el ordenador sobre ese error para identificarlo y resolverlo en el menor tiempo posible. (Rivera, 2015) [10].

De esta forma, podemos decir que al programar estamos desarrollando habilidades de pensamiento computacional y, como consecuencia, mejorando el desempeño en otras áreas disciplinarias.

De acuerdo a (ISTE, NSF & CSTA, 2012) [11], el pensamiento computacional es un proceso de solución de problemas que incluye (pero no se limita a) las siguientes características:

- Formular problemas de manera que permitan usar computadores y otras herramientas para solucionarlos;
- organizar datos de manera lógica y analizarlos;
- representar datos mediante pensamiento algorítmico (una serie de pasos ordenados);
- identificar, analizar e implementar posibles soluciones con el objeto de encontrar la combinación de pasos y recursos más eficiente y efectiva;
- generalizar y transferir ese proceso de solución de problemas a una gran diversidad de éstos.

Además, agrega que estas habilidades se apoyan y acrecientan mediante una serie de disposiciones o actitudes que son dimensiones esenciales del Pensamiento Computacional. Estas disposiciones o actitudes incluyen:

- Confianza en el manejo de la complejidad.
- Persistencia al trabajar con problemas difíciles.
- Tolerancia a la ambigüedad.

- Habilidad para lidiar con problemas no estructurados (*open-ended*).
- Habilidad para comunicarse y trabajar con otros para alcanzar una meta o solución común.

Al respecto, da su opinión Santiago Ceria, director ejecutivo de la Fundación Sadosky, "*Queremos que los alumnos dejen de ser usuarios de un software para empezar a comprender cómo funciona mediante el conocimiento de su estructura y creación. ¿Todos van a ser programadores? No lo creo, cada uno va a elegir su propio camino, pero podrán contar con la capacidad para modelar y abstraer los problemas de la realidad y así encontrar la solución más adecuada*". (Ceria, 2015) [1].

2.1 Evidencias científicas que demuestran los beneficios

Jesús Moreno León en el artículo publicado en el sitio web http://programamos.es/ y titulado "*Evidencias científicas de los beneficios de aprender a programar desde infantil*", realiza un análisis de las evidencias que existen que demuestran los beneficios de enseñar programación en edades tempranas.

> Cuando se comenzó a introducir el lenguaje de programación Logo en la currícula de muchas escuelas de Estados Unidos, distintos grupos de investigadores comenzaron a estudiar qué otras cosas aprendían los alumnos, además de la programación, durante los cursos. Así, en el año 1986, el artículo "*Effects of Logo and CAI environments on cognition and creativity*" (Clements, 1986) [2] explicaba en sus conclusiones que los niños que usaron Logo en educación infantil demostraron mayor **capacidad de atención**, **más autonomía**, y mostraban **un mayor placer por el descubrimiento de nuevos conceptos**.
>
> Otro estudio, "Logo and Geometry" (Clements, Battista y Sarama, 2001), a gran escala de niños que programaban con Logo, demostró que obtuvieron mejores resultados en **pruebas de matemáticas**, **razonamiento** y **resolución de problemas**.

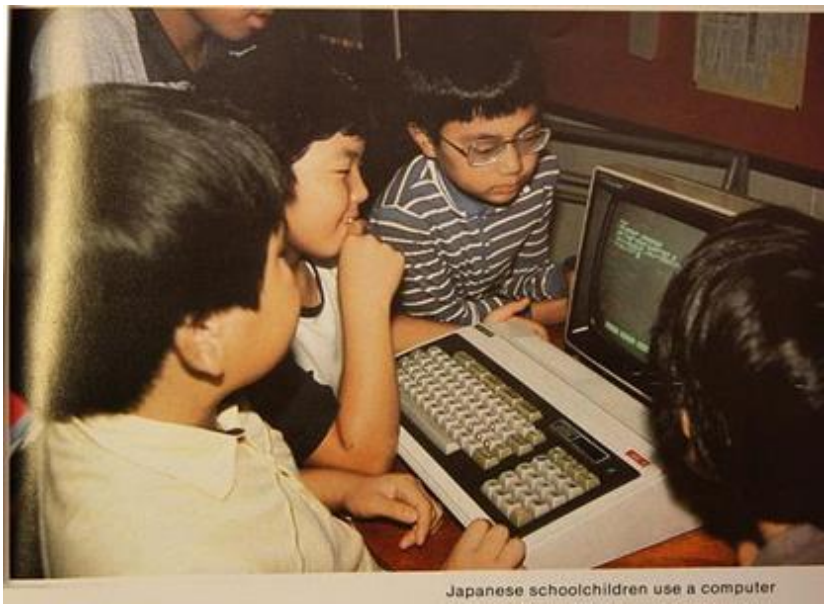

Fig. 1. Escolares japoneses trabajando con un ordenador en los 80. (Sam Howzit. CC-BY, 1980).

Además, estudios más recientes, "*Technology and school change: New lamps for old?*" (Clements & Swaminathan, 2012), han demostrado que aprender a programar tiene un **impacto positivo** en la **creatividad** y **respuesta emocional de niños** con dificultades de aprendizaje, así como en el **desarrollo de las habilidades cognitivas** y **socio-emocionales** (*Effects of computer-assisted instruction and computer programming on cognitive outcomes*).

En relación con las posibilidades que ofrecen las nuevas tecnologías para convertir el aprendizaje en una actividad más social, colaborativa y en red, estudios recientes demuestran que cuando los niños se encuentran trabajando con una computadora hablan **usando más del doble de palabras por minuto** que cuando se encuentran realizando otras actividades sin utilizar dispositivos electrónicos, como jugando con plastilina o pintando "*Early Childhood Education: An International Encyclopedia*". Además, cuando los niños están trabajando con la computadora es más probable que busquen la asistencia y los consejos de otros compañeros, incluso si hay un adulto presente, **incrementando la socialización entre los compañeros** "*Children and computers: New technology – Old concerns*" (Wartella & Jennings, 2000). Incluso cuando se cuenta con una computadora por estudiante, los niños **tienden a formar grupos** mientras trabajan con dispositivos tecnológicos (*The Design of Children's Technology*).

Por último, existen evidencias científicas que demuestran que el alumnado que aprende a programar en edades tempranas tiene **menos estereotipos de género en relación con las carreras STEM** —Ciencias, Tecnología, Ingeniería y Matemáticas— (*Women and Minorities in Science, Technology, En-*

gineering and Mathematics) y **menos reticencias para continuar sus estudios y profesiones en estas disciplinas** (*Gender related to success in science and technology*). (León, 2014) [12].

2.2 Experiencia en el ámbito de la problemática

Durante el segundo semestre del año 2014 realizamos talleres de programación en escuelas primarias, secundarias y en eventos culturales locales, con el objetivo de despertar vocaciones tempranas en carreras Informáticas. Trabajamos con el programa Alice 2.4 porque es una herramienta que introduce a la programación a través del desarrollo de animaciones y videojuegos. Además, el concurso a nivel nacional que lleva a cabo la fundación Sadosky —llamado *Dale Aceptar*[1]—, nos permitió incentivar a los alumnos a que continúen utilizando la herramienta luego de terminar el taller.

Estos cursos nos introdujeron en la problemática planteada de una forma activa, lo que nos permitió pensar en una solución que cumpla con requisitos en el que:

- el alumno se sienta entusiasmado,
- le motive a trabajar con ella,
- donde el docente no tenga dificultades para poder utilizarla en el aula.

3 La herramienta: CÓDIMO

CÓDIMO es el acrónimo de Código, Imaginación y Motivación, originado porque con ella se busca motivar al alumno a que realice actividades cotidianas de sus materias de una forma distinta y que le permita utilizar la imaginación para resolverlas.

Esta aplicación estará basada en Pilas Engine, programa de código abierto, desarrollado por un argentino, que "está dirigido a personas que comienzan a programar [...]", y está "completamente en castellano". (Pilas, 2015) [13].

Pilas Engine utiliza Python como lenguaje de programación. Esto implica que la persona que lo usare deba tener un conocimiento mínimo del inglés para poder entender palabras como *if/then/else*. Por otro lado, los ejercicios que propone para desarrollar, no presentan conceptos educativos, por lo tanto no puede ser utilizada por un docente que no sea especializado en el área, ni aplicado con las actividades diarias de una materia que no sea informática.

Por ello, la solución gramatical de CÓDIMO estará dada por el desarrollo de un pseudo lenguaje en castellano, que será interpretado "en vivo" al lenguaje que comprende Pilas Engine. Por otra parte, mantendrá la estructura gramatical que propone Python, donde se eliminan todos los paréntesis y llaves (los cuales generan complejidad), pero que exige que el código sea indentado correctamente, lo que permite aprender a estructurarlo.

[1] Más información en: http://www.daleaceptar.gob.ar/cms/desafio/de-que-se-trata/.

Por otro lado, CÓDIMO será adaptable a la planificación del docente, y no viceversa, a partir de módulos didácticos que contendrán actividades relacionadas con los temas de la currícula escolar.

Además, el enfoque didáctico de los ejercicios estará orientado a elementos claves tales como la motivación, la relación entre los contenidos y situaciones del mundo real, el trabajo en equipo y la participación activa de los estudiantes. Esto se debe a que creemos que la educación debe basarse en una serie de premisas didácticas que estimulen la curiosidad intelectual, el pensamiento crítico, la creatividad, la autonomía para resolver situaciones y problemas, la toma de decisiones racionales basadas en evidencias, el trabajo colaborativo y cooperativo y el trabajo en actividades interdisciplinarias.

3.1 Proceso de desarrollo

El desarrollo de la aplicación se definió de la siguiente manera:

- Definición del diseño arquitectónico.
- Reunión con los docentes para recabar las posibles actividades a ser implementadas.
- Desarrollo de los prototipos.
- Aceptación de los prototipos a través de la evaluación de docentes competentes en las materias.
- Implementación de los prototipos en forma de módulos.
- Integración de los módulos para la puesta en marcha de la primera versión de la aplicación.

3.1.1 Diseño arquitectónico

El diseño del sistema divide la aplicación en:

- Una Interfaz de Usuario que utilizarán los alumnos y docentes.
- Una capa interna, que se encargará de procesar el código escrito por el usuario y devolver alguna información útil.

Esta división de dos capas permitirá integrar Pilas Engine de una forma mucho más sencilla y así implementar cada una de las actividades de forma separada.

Una actividad es un módulo aislado, el cual tendrá un conjunto limitado de símbolos y reglas semánticas y sintácticas. De esta forma, se podrá desarrollar el pseudo-lenguaje en castellano desde cero, sin tener que implementarlo por completo.

Por lo tanto, es posible aplicar el diseño de software “Iterativo incremental”, donde cada actividad se desarrolla, prueba, integra y se publica en una misma iteración.

3.1.2 Casos de pruebas: Prototipos

Para poder demostrar que es posible utilizar CÓDIMO como herramienta de aprendizaje en materias de distintas disciplinas, desarrollamos dos prototipos, uno para humanidades y otro para exactas.

3.1.2.1 Humanidades: Literatura

Esta actividad busca, a través de un desafío, explicar el mito griego de Teseo y Ariadna, en donde los alumnos tendrán que escapar del laberinto, evitando que el Minotauro los atrape. Habrá tres modos:

- **Modo historia**: (la primera vez que se hace es obligatorio) el juego está acompañado de la narración del mito.
- **Modo reloj de arena**: el alumno tiene un límite de tiempo para escapar del laberinto.
- **Modo eficiencia**: el alumno tiene que resolver el problema en la menor cantidad de instrucciones posible.

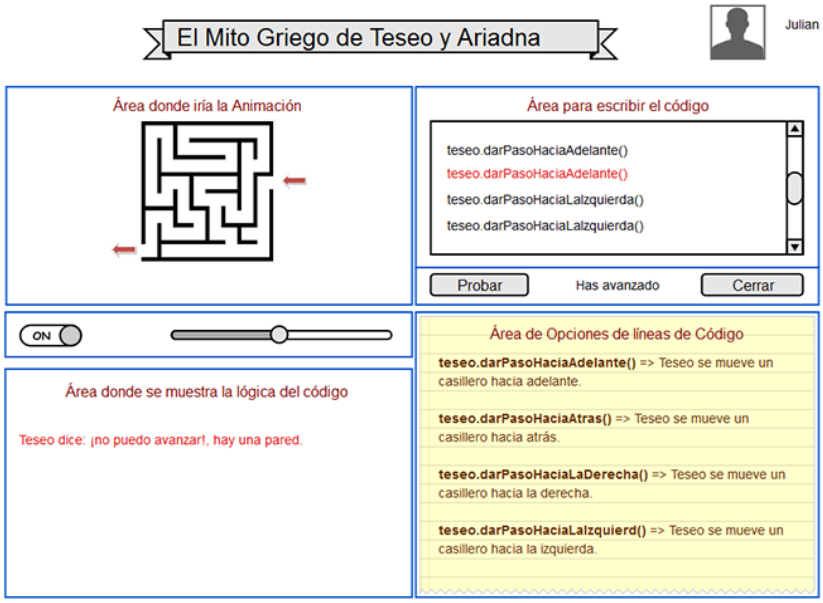

Fig. 2. Prototipo de la vista de la actividad para Literatura.

3.1.2.2 Exactas: Lógica

Un ejercicio aplicable son las compuertas lógicas, dividido en dos etapas:

- La primera está orientada a que el alumno comprenda el funcionamiento interno de una compuerta lógica simple, a través de darle valores a las variables de entrada y salida. La aplicación le muestra la lógica que verifica que el valor de la salida corresponde a los de entrada, y le informa el resultado.

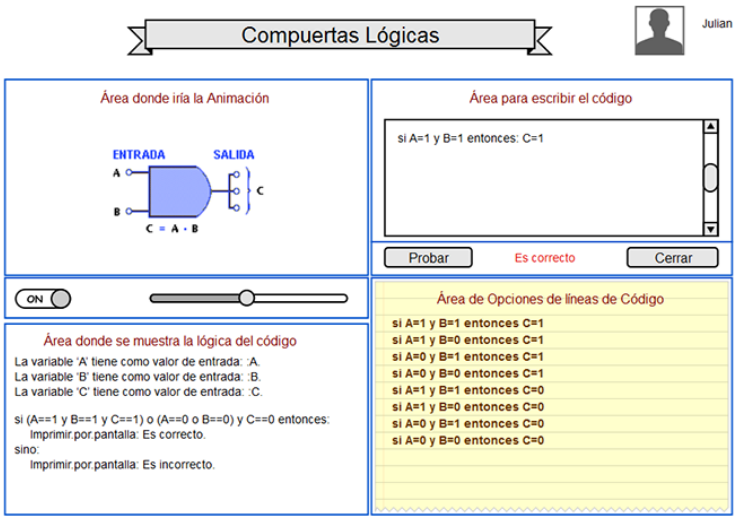

Fig. 3. Prototipo de la vista de la primera actividad para Lógica.

- La segunda está orientada a que el alumno traduzca la imagen de una compuerta lógica compleja a código. La aplicación se encarga de probar todas las posibilidades de la compuerta mediante el código, e informa el resultado. Para poder realizar este tipo de ejercicios, el alumno debe haber aprobado los anteriores.

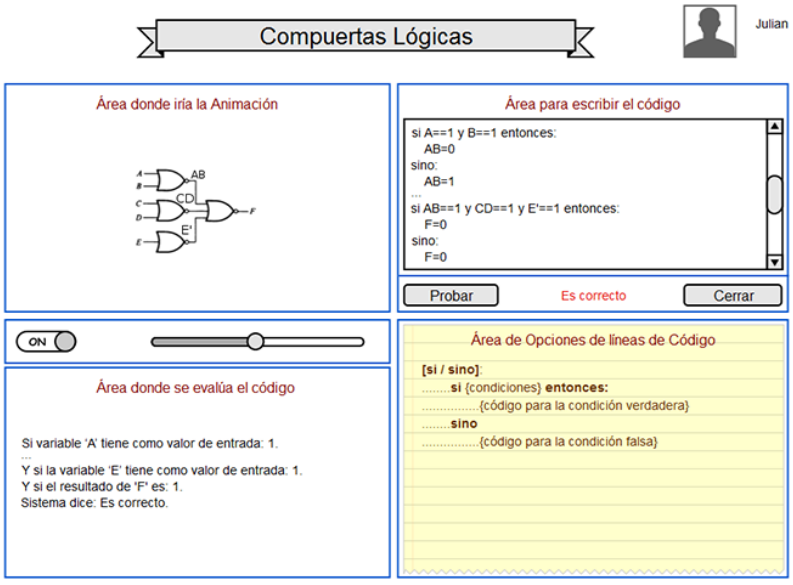

Fig. 4. Prototipo de la vista de la segunda actividad para Lógica.

4 Conclusión

"*El mayor reto para el futuro no es tecnológico sino cultural y educativo. Realmente lo que se necesita es un cambio de mentalidad, de manera que las personas comiencen a ver la programación no solo como un camino hacia un buen empleo sino como una nueva forma de expresión y un nuevo contexto para el aprendizaje*". (Resnick, 2013) [9].

En resumen, podemos decir que al inculcar en el niño o adolescente la programación y el pensamiento computacional, estamos formando personas con habilidades como:

- La abstracción, el razonamiento lógico-secuencial o la modularización.
- El trabajo en equipo o colaborativo.
- Dejar de ser un simple consumidor de tecnología, y pasar a ser un consumidor/productor que tiene conciencia de cómo funciona lo que está usando.
- El uso eficiente de la tecnología más allá del contexto del aula.
- La utilización de la computadora como una herramienta útil, y no únicamente como un dispositivo de entretenimiento.

De acuerdo con Levis nos encontramos frente a la siguiente encrucijada: "*La disyuntiva se plantea entre formar una sociedad de usuarios de la tecnología o una sociedad que incorpore una nueva forma de concebir y crear conocimiento. Una sociedad del conocimiento de los escribas (pocos y garantes de la perpetuación del poder) o una sociedad de letrados (muchos, capaces de generar conocimiento transformador)*". (Levis, 2006) [6].

De esta manera, enseñando a programar estamos formando niños que además de comprender temas matemáticos y computacionales, como variables y condicionales, paralelamente están aprendiendo estrategias para resolver problemas, diseñar proyectos y comunicar ideas. Esas habilidades son útiles no solo para los científicos de la computación sino para todas las personas sin distinción de edad, proveniencia, ocupación o interés.

Referencias

[1] Ceria, S. (2015, 27 de agosto). La nueva alfabetización: enseñarán programación en los colegios públicos. La Nación, recuperado de http://www.lanacion.com.ar/1822530-la-nueva-alfabetizacion-ensenaran-programacion-en-los-colegios-publicos.

[2] Clements, D. H. (1986). Effects of Logo and CAI environments on cognition and creativity. Journal of Educational Psychology, 78(4), 309-318. Recuperado de http://psycnet.apa.org/journals/edu/78/4/309/.

[3] Clements, D. H & Swaminathan, S. (2012). Technology and School Change New Lamps for Old? 275-281. DOI: 10.1080/00094056.1995.10522619.

[4] Didriksson. (2008). Tendencias de la Educación Superior en América Latina y el Caribe. Capítulo 1: CONTEXTO GLOBAL Y REGIONAL DE LA EDUCACIÓN SUPERIOR EN AMÉRICA LATINA Y EL CARIBE. Recuperado de http://www.iesalc.unesco.org.ve/dmdocuments/biblioteca/libros/TENDENCIAS.pdf.

[5] Jonassen, D. (1997). Instructional Design Models for Well-Structure and III Structure Problem-Solving Learning Outcomes. Educational Technology: Research and Development, 45(1), 65-95.

[6] Levis, D. (2006). Alfabetos y saberes: la alfabetización digital. Revista Científica de Comunicación y Educación. 78-82. https://dialnet.unirioja.es/descarga/articulo/1985777.pdf.

[7] López García, J. (2013). Programación de Computadores, un asunto de interés para todos. Eduteka.org. Recuperado de http://www.eduteka.org/articulos/code.

[8] Resnick, M. (2012). Reviving Papert's Dream. Educational Technology.52 (4). 42-46.

[9] Resnick, M. (2013). Aprender a Programar, programar para aprender. Recuperado de http://www.eduteka.org/articulos/codetolearn.

[10] Rivera, N. (2015). Los beneficios psicológicos de la programación. Aparicio, David. Recuperado de http://www.psyciencia.com/2015/02/los-beneficios-psicologicos-de-la-programacion/.

[11] ISTE, NSF & CSTA. (2012). Pensamiento Computacional: Una habilidad de la era digital al alcance de todos. Recuperado de http://www.eduteka.org/modulos/9/272/2082/1.

[12] León Moreno, J. (2014). Evidencias científicas de los beneficios de aprender a programar desde infantil. Programamos.es. Recuperado de http://programamos.es/evidencias-cientificas-de-los-beneficios-de-aprender-a-programar-desde-infantil/.

[13] Pilas Engine (2015). Página principal. Recuperado de http://pilas-engine.com.ar/.